**Java Fundamentals**

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance. Main Concepts - Inheritance, Polymorphism, Abstraction, Encapsulation.

**Data Types in Java**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Default Value** | **Default size** |
| Boolean | false | 1 bit |
| Char | '\u0000' | 2 byte |
| Byte | 0 | 1 byte |
| Short | 0 | 2 byte |
| Int | 0 | 4 byte |
| Long | 0L | 8 byte |
| Float | 0.0f | 4 byte |
| Double | 0.0d | 8 byte |

**Wrapper Classes in Java**

|  |  |
| --- | --- |
| **Primitive Type** | **Wrapper class** |
| Boolean | Boolean |
| Char | Character |
| Byte | Byte |
| Short | Short |
| Int | Integer |
| Long | Long |
| Float | Float |
| Double | Double |

**Operators in Java**

|  |  |  |
| --- | --- | --- |
| **Operator Type** | **Category** | **Precedence** |
| Unary | Postfix | *expr*++ *expr*-- |
| Prefix | ++*expr* --*expr* +*expr* -*expr* ~ ! |
| Arithmetic | Multiplicative | \* / % |
| Additive | + - |
| Shift | Shift | << >> >>> |
| Relational | Comparison | < > <= >= instanceof |
| Equality | == != |
| Bitwise | bitwise AND | & |
| bitwise exclusive OR | ^ |
| bitwise inclusive OR | | |
| Logical | logical AND | && |
| logical OR | || |
| Ternary | Ternary | ? : |
| Assignment | Assignment | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |

**Java Naming Conventions :**

|  |  |
| --- | --- |
| **Name** | **Convention** |
| class name | should start with uppercase letter and be a noun  e.g. String, Color, Button, System, Thread etc. |
| interface name | should start with uppercase letter and be an adjective  e.g. Runnable, Remote, ActionListener etc. |
| method name | should start with lowercase letter and be a verb  e.g. actionPerformed(), main(), print(), println() etc. |
| variable name | should start with lowercase letter e.g. firstName, orderNumber etc. |
| package name | should be in lowercase letter  e.g. java, lang, sql, util etc. |
| constants name | should be in uppercase letter. e.g. RED, YELLOW, MAX\_PRIORITY etc. |

**Object vs Class**

|  |  |
| --- | --- |
| **Object** | **Class** |
| Object is an **instance** of a class. | Class is a **blueprint or template** from which objects are created. |
| Object is a **real world entity** such as pen, laptop, mobile, bed, keyboard, mouse, chair etc. | Class is a **group of similar objects**. |
| Object is a **physical** entity. | Class is a **logical** entity. |
| Object is created through **new keyword** mainly e.g. Student s1=new Student(); | Class is declared using **class keyword** e.g. class Student{} |
| Object is created **many times** as per requirement. | Class is declared **once**. |
| Object **allocates memory when it is created**. | Class **doesn't allocated memory when it is created**. |
| There are **many ways to create object** like new keyword, newInstance() method, clone() method, factory method & deserialization. | There is only **one way to define class** in java using class keyword. |

**Constructors vs Methods**

|  |  |
| --- | --- |
| **Java Constructor** | **Java Method** |
| Constructor is used to initialize the state of an object. | Method is used to expose behaviour of an object. |
| Constructor must not have return type. | Method must have return type. |
| Constructor is invoked implicitly. | Method is invoked explicitly. |
| Compiler provides a default constructor if you don't have any constructor. | Method is not provided by compiler in any case. |
| Constructor name must be same as the class name. | Method name may or may not be same as class name. |